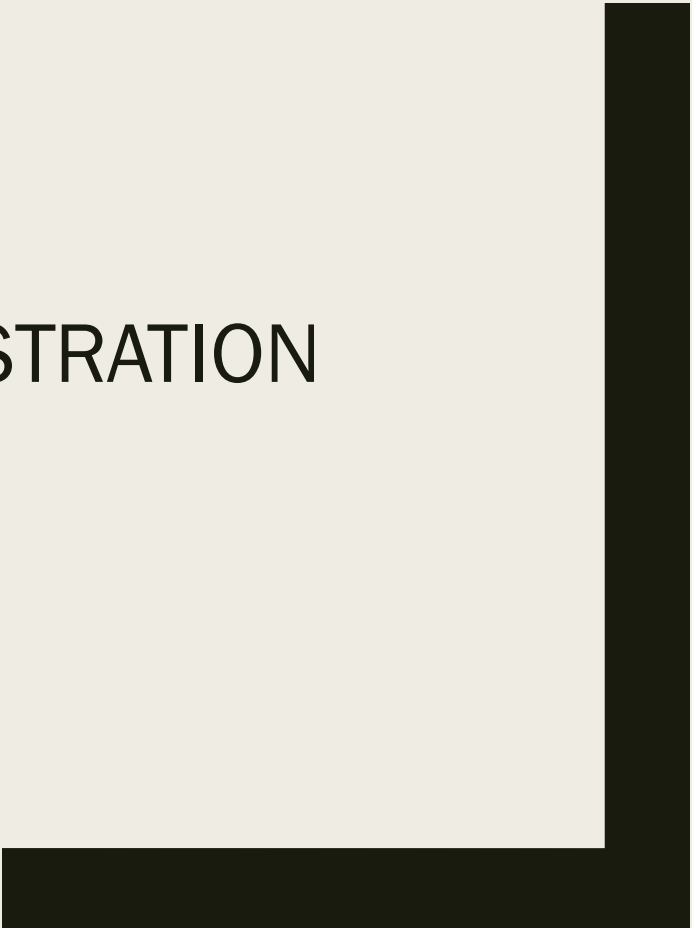




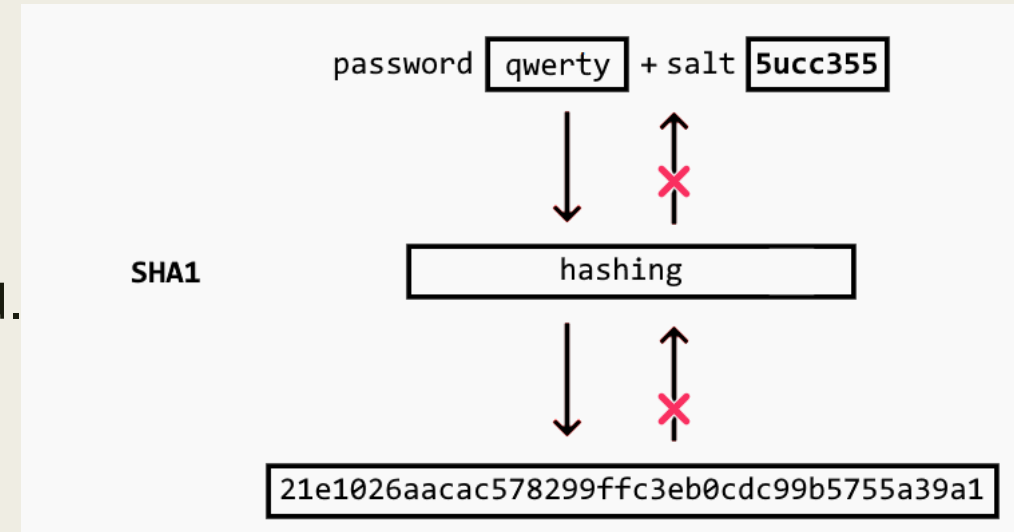
RAINBOW TABLE ATTACK DEMONSTRATION

Team Members: Engy Ashraf Samy 20236017
Hana Ayman Mohamed 20236116
Jana Tamer Mohamed 20236026
Rawan Hassan Mohamed 20236038
Logyn Hossam Eldeen 20236075



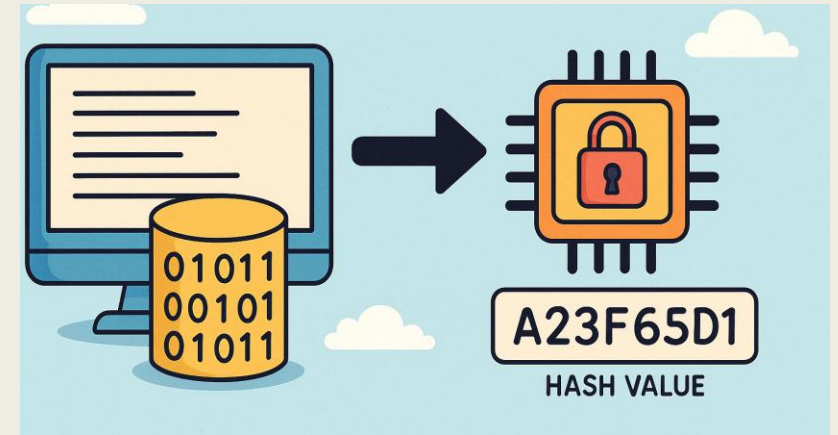
What Is Password Hashing ?

- Password hashing converts a password into a fixed-length value.
- The output is called a **hash**.
- The same password always produces the same hash.
- Hashing is a **one-way function** and cannot be reversed.
- Hash functions are deterministic.
- Hash functions are designed to be computationally efficient but secure.
- → Passwords are stored as hashes, not as plain text



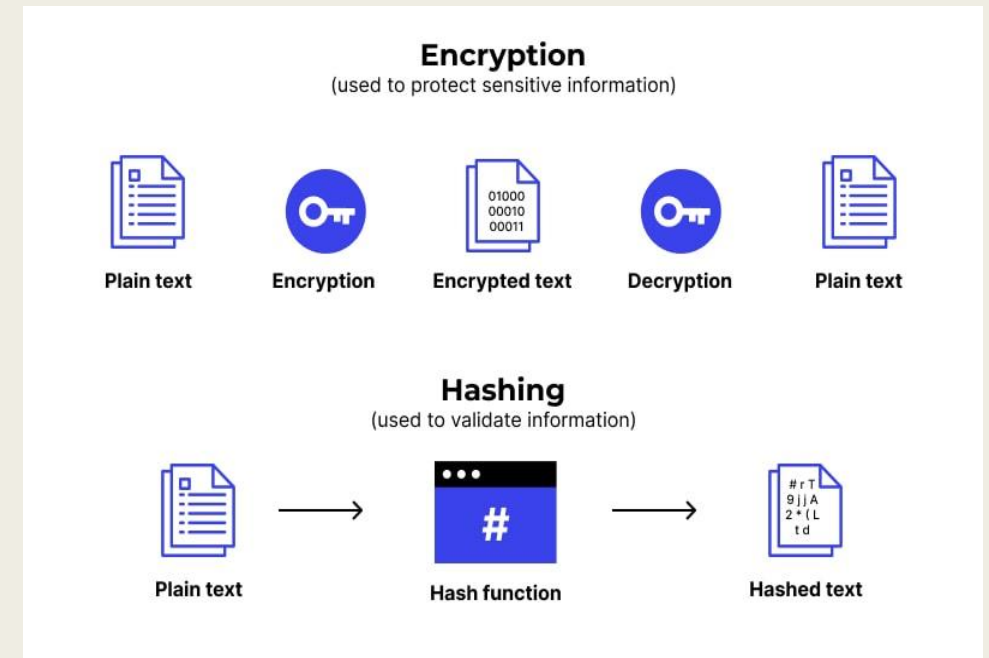
Why Systems Use Hashing ?

- Protects passwords if the database is compromised.
- Prevents attackers from reading original passwords.
- Prevents system administrators from knowing user passwords.
- Reduces damage caused by data breaches.
- Essential for secure authentication systems.



Hashing vs Encryption

- **Hashing:**
- One-way process.
- Cannot be reversed.
- Used for password storage.
- Encryption requires secure key management, hashing does not.
- **Encryption:**
- Two-way process.
- Can be reversed using a key.
- Used for data confidentiality.
- → Passwords must be hashed, not encrypted.



What Is a Rainbow Table Attack ?

- A rainbow table is a precomputed database of passwords and hashes.
- Attackers match stolen hashes against the table.
- No need to guess passwords during the attack.
- Faster than brute force attacks.
- Targets weak hashing implementations.
- Rainbow tables rely on the predictability of hash outputs.



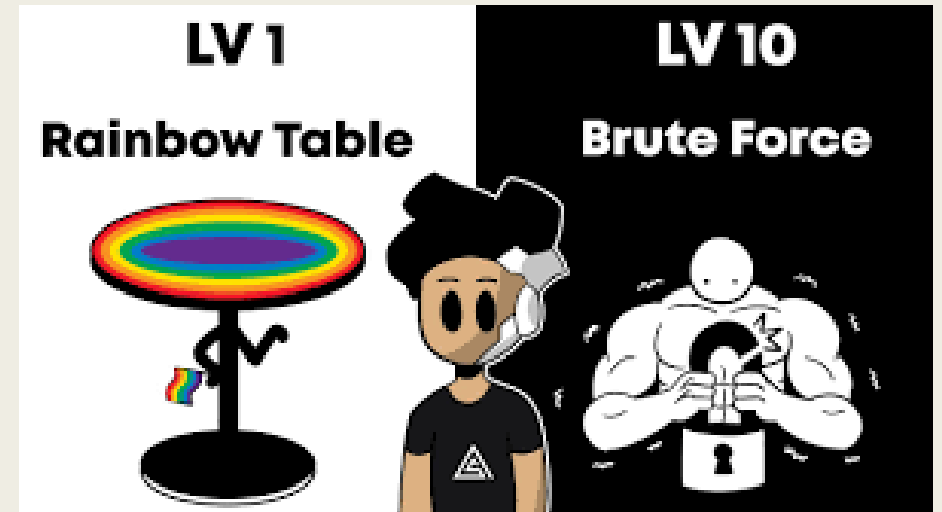
Why Rainbow Table Attacks Are Effective

- Hashes are computed in advance.
- Saves time during the attack phase.
- Extremely fast compared to brute force.
- Effective against common and weak passwords.
- Works well on unsalted hash databases.
- Attack efficiency increases with fast hash algorithms.



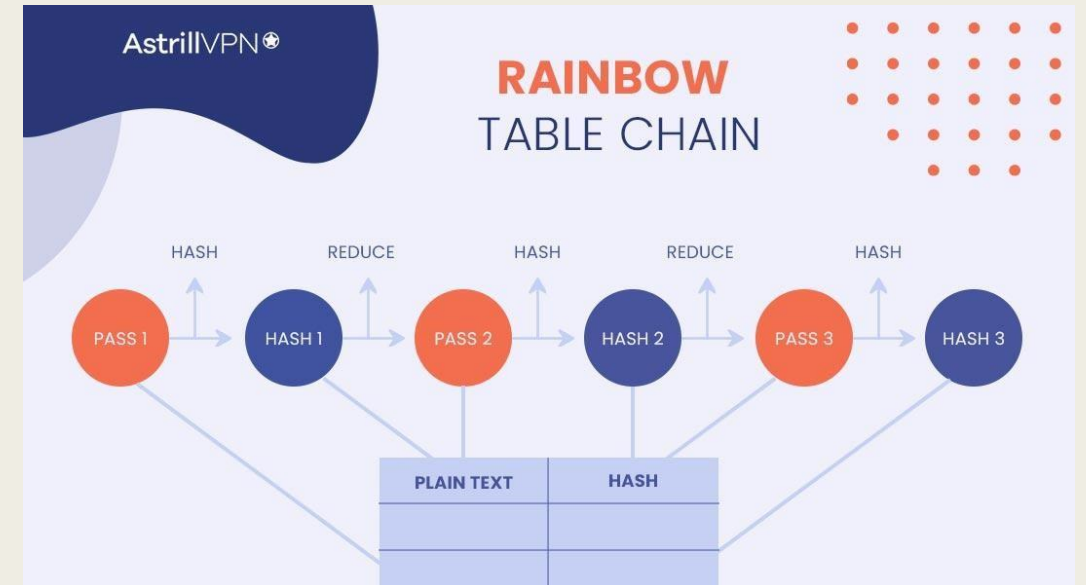
Brute Force vs Rainbow Table

- **Brute Force Attack:**
- Tries all possible password combinations.
- Very slow for large password spaces.
- High CPU usage.
- No precomputation.
- **Rainbow Table Attack:**
- Uses precomputed hash tables.
- Very fast lookup.
- Requires large storage space.
- Limited to known password lists.
- Precomputation phase performed once.



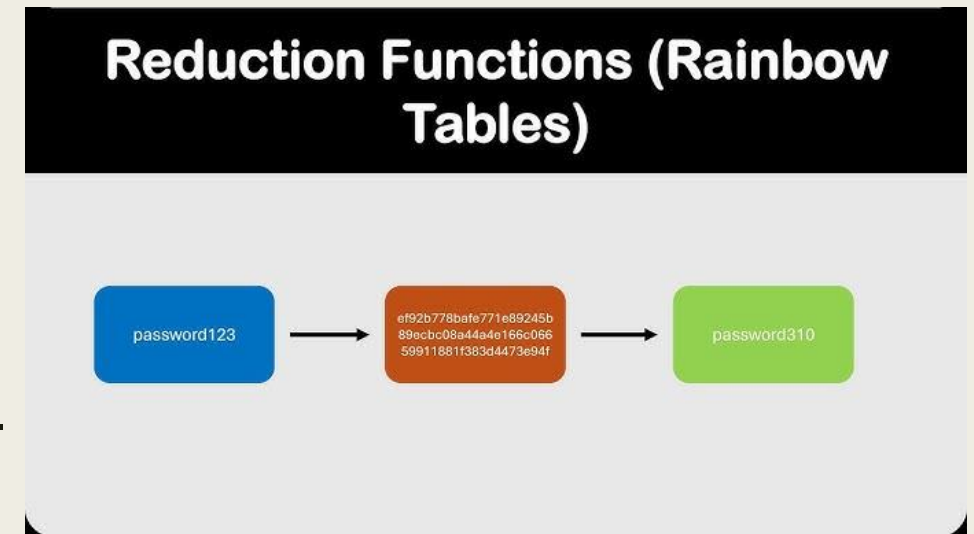
How Rainbow Tables Work

- Generate a list of common passwords.
- Apply a hash function to each password.
- Store hash–password relationships.
- Use reduction functions to reduce storage.
- Compare stolen hashes with the table.



Reduction Function Concept

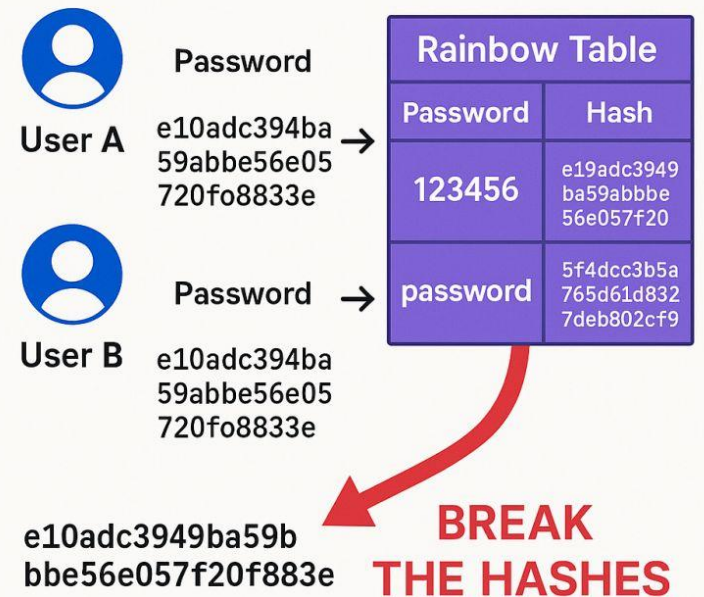
- Reduction functions map hash values back to possible passwords.
- They do not reverse the hash.
- Used to form chains in rainbow tables.
- Reduce storage requirements.
- Improve efficiency of large tables.
- Reduction functions are not cryptographic functions.



Attack Scenario

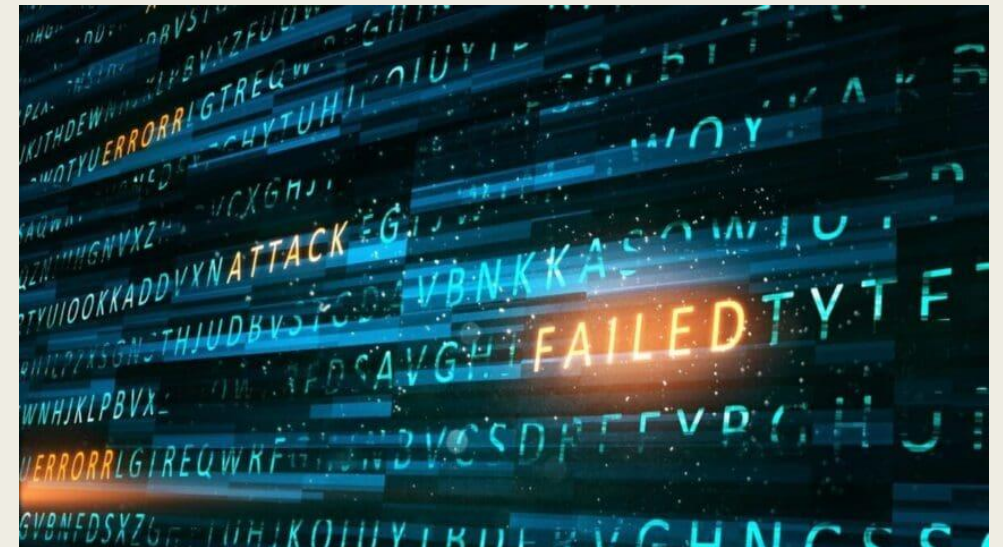
- Attacker gains access to a database of hashed passwords.
- Hashes are extracted from the database.
- Each hash is searched in the rainbow table.
- If a match exists, the password is revealed.
- Attack succeeds without guessing.

RAINBOW TABLE ATTACK



When Rainbow Tables Fail

- Strong and complex passwords.
- Long password length.
- Use of unique salts.
- Use of slow hashing algorithms.
- Modern password protection techniques.
- Use of key stretching techniques.



Real-World Impact

- Older systems using MD5 or SHA-1.
- Databases without salting.
- Legacy systems with weak security.
- Past data breaches exposed hashed passwords.
- Motivated stronger password policies.
- Led to security standards recommending salted hashing.



Tools Used in the Demo

- Python programming language.
- hashlib cryptographic library.
- Simple password wordlist.
- Command line or IDE environment.
- Console-based execution.
- MD5 used for educational purposes only.

Live Demonstration Overview

- Build a small rainbow table.
- Hash a weak password.
- Crack the password successfully.
- Add a salt to the hash.
- Demonstrate attack failure.
- Show why MD5 is insecure and unsalted hashes are vulnerable.

Ethical Note

- Demonstration is for educational purposes only.
- Understanding attacks helps build better defenses.
- Unauthorized password cracking is illegal.
- Ethical hacking requires permission.



Defense Against Rainbow Tables

- Add a unique salt to each password.
- Use slow hashing algorithms:
 - *bcrypt*
 - *scrypt*
 - *Argon2*
- Enforce strong password rules.
- Avoid outdated hash algorithms.
- MD5 and SHA-1 are deprecated for password storage.



Conclusion

- Rainbow tables exploit weak hashing practices.
- Precomputation makes attacks extremely fast.
- Unsalted hashes are highly vulnerable.
- Salting and modern hashing defeat the attack.
- Security depends on correct implementation, not just algorithms.
- Secure password storage is essential.